UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/594,799	09/28/2006	Zach Yoav	42P23148	8317

45209 7590 10/20/2010
INTEL/BSTZ
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
TANG, KENNETH	

ART UNIT	PAPER NUMBER
2196	

MAIL DATE	DELIVERY MODE
10/20/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/594,799	**Applicant(s)** YOAV ET AL.
	Examiner KENNETH TANG	**Art Unit** 2195

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _28 September 2006 and 15 June 2008_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-21_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-21_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _6/15/08_ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-21 are presented for examination.

Claim Objections

2. **Claims 1 and 19 are objected to because of the following informalities:**

- In claim 1, line 2, "beginning initialization a foreign thread" is grammatically incorrect.
- In claims 1 and 19, the term "capable of" invokes the intended use, which does not positively recite the claims. Language that suggests or makes optional but does not require steps to be performed or does not limit a claim to a particular structure does not limit the scope of a claim or claim limitation (MPEP 2106). It is advised that the claims 1 and 19 positively recite the claim language.

Appropriate correction is required.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. **Claims 15-18 are directed to non-statutory subject matter.**

4. As per claim 15, it is directed to a system that is software, per se. Such a claim that is directed to a software system fails to fall under one of the four statutory categories of invention under 35 USC 101. Claims 16-18 are also rejected for being dependent on rejected claim 15 and for failing to cure its deficiencies.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

5. **Claims 1-5 are rejected under 35 U.S.C. 102(e) as being anticipated by Kissell (US 2005/0120194 A1).**

6. As per claim 1, Kissell teaches a computer implemented method comprising:

beginning initialization a first thread from a second context, wherein the first thread is <u>capable of</u> being executed on a first context and the second context (Abstract; [0049]);

suspending the initialization of the first thread at a position within the second context (suspending occurs from the interrupt of the context switching, etc.) ([0007]; [0008]);

creating a second thread based on the position in the second context ([0033]; [0042]; [0043]); and

completing the initialization of the first thread continuing from the position in the second context ([0033]; [0042]; [0043]).

7. As per claim 2, Kissell teaches wherein the beginning initialization of the first thread includes allocating per-thread context resources (Fig. 2, item 226; [0030]; [0031]).

8. As per claim 3, Kissell teaches wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread (suspending occurs from the interrupt of the context switching, etc.) ([0007]; [0008]).

9. As per claim 4, Kissell teaches wherein the second context is the platform-independent code to be executed on a host platform (JAVA, etc.) ([0068]; [0069]).

10. As per claim 5, Kissell teaches wherein the second context is a host platform that supports multiple instruction set architectures (ISA) (MIPS32 or MIPS64 Instruction Set Architectures, etc.) ([0031]).

11. Claims 15-19 are rejected under 35 U.S.C. 102(e) as being anticipated by Wang et al. (hereinafter Wang) (US 2006/0184920 A1).

12. As per claim 15, Wang teaches a computer system for managing thread resource comprising:

a first multithreaded programming environment ([0042]; Abstract);

a second multithreaded programming environment ([0042]; Abstract);

a multithreaded program including a first thread and a second thread ([0042]);

a host platform ([0002]; [0006]; [0007]); and

a dynamic binary translator to manage and support the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment ([0029]; Fig. 1, item 112).

13. As per claim 16, Wang teaches further comprises a first component to provide a communication interface between the dynamic binary translator and the multithread programming environment (Fig. 1, item 109).

14. As per claim 17, Wang teaches further comprises a first thread library and a second thread library ([0042]).

15. As per claim 18, Wang teaches further comprises a second component to intercept service requests from the multithreaded program ([0040]).

16. As per claim 19, Wang teaches a computer system for managing thread resource comprising (see Abstract):

a random accessed memory (RAM 1106, Fig. 11);

a first processor capable of executing multithreaded programs stored in the random accessed memory (Processor 1102 executing multithreaded programs stored in RAM 1106, Fig. 11);

a multithreaded program to be executed on a second processor ([0042]; Fig. 11, items 1102, 1144, etc.); and

a program to transparently initialize and create a thread included in the multithreaded program in an environment supported by the first processor, to be executed on the second processor ([0042]; Fig. 11, items 1102, 1144, etc.).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. Claims 6-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kissell (US 2005/0120194 A1) in view of Venstermans et al. (hereinafter Venstermans) ("64-bit versus 32-bit Virtual Machines for Java", September, 15, 2005).

18. As per claim 6, Kissell does not expressly teach wherein completing the initialization of the first thread includes executing an application programming interface that makes an operating system request to create the first thread in the second context. However, Venstermans teaches wherein completing the initialization of the first thread includes executing an application programming interface that makes an operating system request to create the first thread in the second context (page 5, 2nd paragraph). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Kissell and Venstermans because it would provide the predicted result of a means to communicate between two different operating environments (Venstermans – Abstract).

19. As per claim 7, Kissell does not expressly teach wherein the first context is an IA-32 multithreaded program and the second context is an IA-64 multithreaded program. However, Venstermans teaches wherein the first context is an IA-32 multithreaded program and the second context is an IA-64 multithreaded program (see Abstract). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Kissell

and Venstermans because it would provide the predicted result of a means to communicate between two different operating environments (Venstermans – Abstract).

20. As per claim 8, it is rejected for similar reasons as stated in the rejection of claim 7.

21. As per claim 9, it is rejected for similar reasons as stated in the rejection of claim 1. However, Bissell does not expressly teach a foreign thread is to be executed on a foreign platform. But Venstermans teaches a different thread to be executed on a different platform (Abstract). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Kissell and Venstermans because it would provide the predicted result of a means to communicate between two different operating environments (Venstermans – Abstract).

22. As per claim 10, Kissell (Fig. 2, item 226; [0030]; [0031]) and Venstermans (Abstract) teaches wherein the beginning initialization of the foreign thread includes allocating per-thread context resources.

23. As per claim 11, Kissell ([0007]; [0008]) and Venstermans (Abstract) teaches wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread.

24. As per claim 12, Kissell ([0068]; [0069]) and Venstermans (Abstract) teaches wherein the host platform supports platform-independent code.

25. As per claim 13, Kissell ([0031]) and Venstermans (Abstract) teaches wherein the host platform supports multiple instruction set architectures (ISA).

26. As per claim 14, Venstermans teaches wherein the foreign platform is a IS-32 platform and the host platform is a IS-64 platform (Abstract).

27. As per claim 15, Kissell teaches a computer system for managing thread resource comprising:

a first multithreaded programming environment ([0031]);

a second multithreaded programming environment ([0031]);

a multithreaded program including a first thread and a second thread ([0025]; Abstract);

a host platform [0068]; [0069]; and

Kissell does not expressly teach a dynamic binary translator to manage and support the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment. However, Venstermans teaches a computer system that utilizes a Java Virtual Machine to provide a dynamic binary translator to manage and support the first thread for the first multithreaded programming environment (32-bit mode, etc.) to be executed in the second multithreaded programming environment (64-bit mode, etc.). (see Abstract). It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Kissell such that it would include the feature of a dynamic binary translator to manage and support the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment, as taught and suggested in Venstermans. The suggestion/motivation for doing so would have been to provide the predicted result of taking advantage of the Java language's benefit of it platform independence, making it useful in a lot of technologies ranging from embedded devices to high-performance systems (Abstract).

28. As per claim 16, Venstermans teaches further comprises a first component to provide a communication interface between the dynamic binary translator and the multithread programming environment (page 5, 2nd paragraph).

29. As per claim 17, Kissell ([0007]; [0017]) in view of Venstermans (page 5, 2nd paragraph) teaches further comprises a first thread library and a second thread library.

30. As per claim 18, Kissell teaches further comprises a second component to intercept service requests from the multithreaded program (interrupt of the context switching, etc.) ([0007]; [0008])..

31. Claims 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kissell (US 2005/0120194 A1).

32. As per claim 19, Kissell teaches a computer system for managing thread resource comprising:

a random accessed memory (System memory 108) (Fig. 1);

a first processor capable of executing multithreaded programs stored in the random accessed memory (multithreaded microprocessor 102) (Fig. 1);

a multithreaded program to be executed; and

a program to transparently initialize and create a thread included in the multithreaded program in an environment supported by the first processor.

33. Kissell does not explicitly disclose a second processor. However, one of ordinary skill in the art would find it obvious that Kissell's multithreaded microprocessor is capable of containing multiple logical processors. The suggestion/motivation for executing on a plurality of logical processors would be to provide the predicted result of increasing parallelism.

34. As per claim 20, Kissell teaches wherein the program further allocates per-thread context resources (Fig. 2, item 226; [0030]; [0031]).

35. As per claim 21, Kissell teaches wherein the program initializes and creates the thread transparently by associating the allocated per-thread context resource between the environment supported by the first processor (Fig. 2, item 226; [0030]; [0031]).

36. Claims 1-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chen et al. (hereinafter Chen) ("Java JNI Bridge: A Framework for Mixed Native ISA Execution," March 26-29, 2006).

37. As per claim 1, Chen teaches a computer implemented method comprising:

beginning initialization a first thread from a second context, wherein the first thread is capable of being executed on a first context and the second context (Section 3.2.5; Section 3.3.3);

creating a second thread based on the position in the second context (parent thread spawns a child thread) (Section 3.2.5); and

completing the initialization of the first thread continuing from the position in the second context (Section 3.2.5).

38. Chen does not expressly disclose suspending the initialization of the first thread at a position within the second context. However, Kissell teaches a computer implemented method comprising: beginning initialization a first thread from a second context, wherein the first thread is capable of being executed on a first context and the second context (Abstract; [0049]); suspending the initialization of the first thread at a position within the second context (suspending occurs from the interrupt of the context switching, etc.) ([0007]; [0008]); creating a second thread based on the position in the second context ([0033]; [0042]; [0043]); and completing the initialization of the first thread continuing from the position in the second context ([0033]; [0042]; [0043]). It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Chen's thread processing such that it would include the feature of suspending the initialization of the first thread at a position within the second context, as taught and suggested in Kissell. The suggestion/motivation for doing so would have been to provide the predicted result of making the forking from parent thread to child thread more lightweight and efficient, and executable within a single processor clock cycle (Kissell – [0043]).

39. As per claim 2, Chen teaches wherein the beginning initialization of the first thread includes allocating per-thread context resources (Section 3.2.5).

40. As per claim 3, Kissell teaches wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread (suspending occurs from the interrupt of the context switching, etc.) ([0007]; [0008]).

41. As per claim 4, Chen teaches wherein the second context is the platform-independent code to be executed on a host platform (Java, etc.) (Abstract).

42. As per claim 5, Chen teaches wherein the second context is a host platform that supports multiple instruction set architectures (ISA) (Mixed ISA, etc.) (Abstract).

43. As per claim 6, Chen teaches wherein completing the initialization of the first thread includes executing an application programming interface that makes an operating system request to create the first thread in the second context (Section 3.2).

44. As per claim 7, Chen teaches wherein the first context is an IA-32 multithreaded program and the second context is an IA-64 multithreaded program (2nd paragraph of 1. Introduction; Section 2.3).

45. As per claim 8, Chen teaches wherein the first context is an IA-32 platform and the second context is an IA-64 platform (2nd paragraph of 1. Introduction; Section 2.3).

46. As per claim 9, it is rejected for the same reasons as stated in the rejection of claim 1. In addition, the foreign thread represents a thread on a different/outside platform, as taught in Chen.

47. As per claim 10, Chen teaches wherein the beginning initialization of the foreign thread includes allocating per-thread context resources (Section 3.2.5).

48. As per claim 11, it is rejected for the same reasons as stated in the rejection of claim 3.

49. As per claim 12, Chen teaches wherein the host platform supports platform-independent code (Java, etc.) (Abstract).

50. As per claim 13, Chen teaches wherein the host platform supports multiple instruction set architectures (ISA) (see Abstract).

51. As per claim 14, Chen teaches wherein the foreign platform is a IS-32 platform and the host platform is a IS-64 platform (2nd paragraph of 1. Introduction; Section 2.3).

52. As per claim 15, Chen teaches a computer system for managing thread resource (Abstract) comprising:

a first multithreaded programming environment (Figure 1; Section 3.2.5);

a second multithreaded programming environment (Figure 1; Abstract);

a multithreaded program including a first thread and a second thread (Figure 1; Abstract);

a host platform (Figure 1; 1. Introduction); and

a dynamic binary translator to manage and support the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment (Figure 1; Section 2.2).

53. As per claim 16, Chen teaches further comprises a first component to provide a communication interface between the dynamic binary translator and the multithread programming environment (Figure 1).

54. As per claim 17, Chen teaches further comprises a first thread library and a second thread library (Section 3.3).

55. As per claim 18, Chen teaches further comprises a second component to intercept service requests from the multithreaded program (Signal handler, etc.) (Figure 2).

56. As per claim 19, Chen teaches a computer system for managing thread resource (see Abstract) comprising:

a random accessed memory (inherent in the computer system) (Section 1);

a first processor <u>capable of</u> executing multithreaded programs stored in the random accessed memory (Section 1);

a multithreaded program to be executed on a second processor (Abstract); and

a program to transparently initialize and create a thread included in the multithreaded program in an environment supported by the first processor, to be executed on the second processor (Section 3.3).

57. As per claim 20, Chen teaches wherein the program further allocates per-thread context resources (Section 3.2.5).

58. As per claim 21, Chen teaches wherein the program initializes and creates the thread transparently by associating the allocated per-thread context resource between the environment supported by the first processor (Section 3.3; Section 3.2.5).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to KENNETH TANG whose telephone number is (571)272-3772. The examiner can normally be reached on 9:00AM - 5:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Emerson Puente can be reached on (571) 272-3652. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

/Kenneth Tang/
Examiner, Art Unit 2195